Theses and Dissertations                                          1. Thesis and Dissertation Collection, all items

2008-12

# Development and testing of a new area search model with partially overlapping target and searcher patrol areas

Kim, Gi Young

Monterey  California. Naval Postgraduate School

http://hdl.handle.net/10945/3843

# NAVAL
# POSTGRADUATE
# SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**DEVELOPMENT AND TESTING OF A NEW AREA SEARCH MODEL WITH PARTIALLY OVERLAPPING TARGET AND SEARCHER PATROL AREAS**

by

Gi Young Kim

December 2008

Thesis Advisor:    James N. Eagle
Second Reader:    Sung Jin Kang

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | *Form Approved OMB No. 0704-0188* |
|---|---|---|

| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE<br>December 2008 | 3. REPORT TYPE AND DATES COVERED<br>Master's Thesis | |
|---|---|---|---|
| 4. TITLE AND SUBTITLE Development and Testing of a New Area Search Model with Partially Overlapping Target and Searcher Patrol Areas | | 5. FUNDING NUMBERS | |
| 6. AUTHOR(S) Gi Young Kim | | | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | | 8. PERFORMING ORGANIZATION REPORT NUMBER | |
| 9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>N/A | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER | |

| 11. SUPPLEMENTARY NOTES  The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. |
|---|

| 12a. DISTRIBUTION / AVAILABILITY STATEMENT<br>Approved for public release; Distribution is unlimited | 12b. DISTRIBUTION CODE |
|---|---|

**13. ABSTRACT (maximum 200 words)**

In this study, the author uses a MATLAB simulation to develop and test a generalization of the traditional Random Search model which allows both the searcher and target to move and to be in different, but overlapping, areas. Also the best evasion speed for a randomly moving target against a Systematic Search is studied.

| 14. SUBJECT TERMS Random Search, Exhaustive Search, Systematic Search, Search and Detection, MATLAB Simulation | | 15. NUMBER OF PAGES<br>63 |
|---|---|---|
| | | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT<br>Unclassified | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>Unclassified | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>Unclassified | 20. LIMITATION OF ABSTRACT<br>UU |
|---|---|---|---|

i

THIS PAGE INTENTIONALLY LEFT BLANK

**DEVELOPMENT AND TESTING OF A NEW AREA SEARCH MODEL WITH PARTIALLY OVERLAPPING TARGET AND SEARCHER PATROL AREAS**

Gi Young Kim
Captain, Republic of Korea Air Force
B.S., Republic of Korea Air Force Academy, 1998

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS ANALYSIS**

from the

**NAVAL POSTGRADUATE SCHOOL**
**December 2008**

Author:          Gi Young Kim

Approved by:     James N. Eagle
                 Thesis Advisor

                 Sung Jin Kang
                 Second Reader

                 James N. Eagle
                 Chairman, Department of Operations Research

iii

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

In this study, the author uses a MATLAB simulation to develop and test a generalization of the traditional Random Search model which allows both the searcher and target to move and to be in different, but overlapping, areas. Also the best evasion speed for a randomly moving target against a Systematic Search is studied.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# EXECUTIVE SUMMARY

In this study, the author uses a MATLAB simulation to develop and test a generalization of the traditional Random Search model which allows both the searcher and target to move and to be in different, but overlapping, areas. Also the best evasion speed for a randomly moving target against a Systematic Search is studied.

The new generalized Random Search formula is,

$$F_T(t) = \alpha F_{Tslow}(t) + (1-\alpha)F_{Tfast}(t) \quad , \text{where}$$

$$\alpha = \exp(-\xi\sqrt{t}) \, ,$$

$$\xi = \frac{A_{st}U}{\sqrt{A_s A_t RV}} \, ,$$

$$F_{Tslow}(t) = (A_{st}/A_t)(1-\exp(-2R\tilde{V}t/A_s)) \, ,$$

$$F_{Tfast}(t) = 1-\exp(-2R\tilde{V}(A_{st}/A_t)t/A_s) \, , \text{ and}$$

$$\tilde{V} = \text{ mean relative speed between searcher and target.}$$

With regard to the best evasion speed against a Systematic searcher, extensive simulation suggests that a 5%-20% of searcher speed is the best speed for the target.

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. BACKGROUND

Random Search is a popular model for area search because it is mathematically simple and it provides a conservative, lower bound on the probability of detecting a stationary target with Systematic Search. However, the Random Search model also has significant limitations. In particular, it assumes that the searcher and target are contained in the same area and that the target is stationary. We address these model limitations in this thesis.

## B. RESEARCH QUESTIONS

This thesis will address the following two questions:

1. How can the Random Search model be generalized for situations where the searcher and target areas are not coincident and both the searcher and target are moving?

2. What is the best speed for a randomly moving target to evade a searcher conducting a Systematic Search?

## C. THESIS ORGANIZATION

The thesis is organized in the following manner:

- Chapter II reviews the search models of Exhaustive and Random Search.

- Chapter III presents a MATLAB simulation of Random Search.

- Chapter IV develops the extended Random Search model where searcher and target areas are not coincident and both the searcher and target are moving.

- Chapter V investigates the optimal speed for a target evading a Systematic Search.

- Chapter VI summarizes the results of simulations and recommends future studies.

# II. OVERVIEW OF RANDOM SEARCH THEORY[1]

## A.    EXHAUSTIVE SEARCH

### 1.    Characteristics of Exhaustive Search

Exhaustive Search is an idealized search model which assumes a stationary target and a 'perfect search', meaning no search overlap, no search effort placed outside the target area, and all of target area is completely covered by the search sensor. The searcher has a 'cookie-cutter sensor' with range $R$, sometimes called a definite range law sensor. Such a sensor always detects a target within a specified range $R$, and never detects targets beyond that range. That is:

$$P(\text{detection})=P_d = \begin{cases} 1, \text{ range} \leq R \\ 0, \text{ range} > R . \end{cases} \qquad \textbf{(1)}$$

Because of its optimistic area coverage assumptions, Exhaustive Search is generally assumed to provide an upper bound on the performance of realistic search.

### 2.    Exhaustive Search for a Uniformly Distributed Target

Assume that a searcher conducts Exhaustive Search over the search area $A$ ( $\text{mn}^2$ ) with a cookie-cutter sensor of range $R$ ( nm ), speed $V$ ( nm/hour ), and sweep rate $2VR$ ( nm/hour ). The

---

[1] The references for this section are the unpublished lecture notes of Professor James N. Eagle and Professor Alan R. Washburn's 'Search and Detection' book (2002).

time of detection in the random variable $T$ ( hour ), which is uniformly distributed between time 0 and $A/(2VR)$. That is, $T \sim U [\, 0, A/(2VR) \,]$.

The probability density function and cumulative distribution function for random variable $T$ are:

$$f_T(t) = \begin{cases} (2VR)/A, \ t \in [0, A/(2VR)] \\ 0, \text{ otherwise} \end{cases}$$

$$F_T(t) = \begin{cases} 0, \ t < 0 \\ (2VRt)/A, \ t \in [0, A/(2VR)] \\ 1, \ t > A/(2VR) \end{cases} \qquad (2)$$

The mean time to detection is given by the expected value of $T$:

$$E(T) = A/(4VR) . \qquad (3)$$
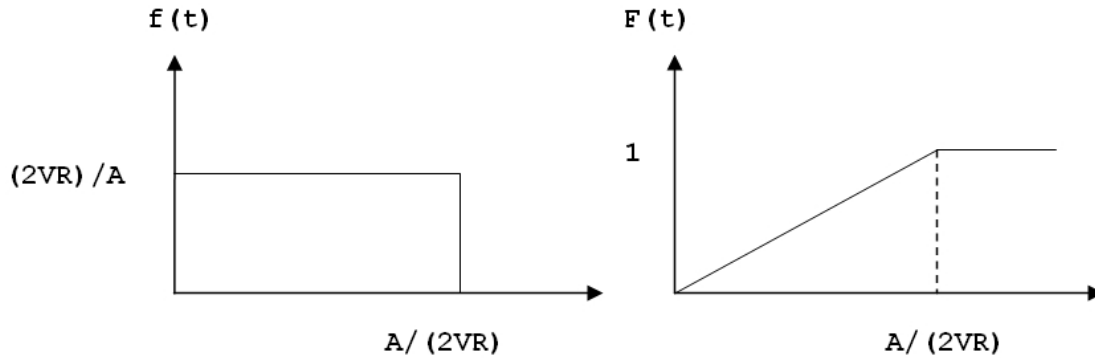


Figure 1.    Density and CDF for the Time of Initial Detection $T$ with Exhaustive Search.

B.    CONTINUOUS SEARCH

    1.    Model Assumption and Definitions

    •    $\gamma(t)$ = detection rate at time $t$ (units: 1/time).
        Detection rate has two equivalent interpretations:

4

*i.* $P(\text{detection in } [t, t+\Delta t]) = \gamma(t)\Delta t + o(\Delta t)$

$\approx \gamma(t)\Delta t, \text{ for small } \Delta t$ **(4)**

Note: $o(\Delta t)$ is a function of $\Delta t$ which goes to 0 faster than linearly. That is:

$$\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$$

*ii.* $E(\text{\# detections in } [t, t+\Delta t]) = \int_{t}^{t+\Delta t} \gamma(s)ds$

$\approx \gamma(t)\Delta t, \text{ for small } \Delta t$ **(5)**

- $T$ is the random time of initial detection, and we define $F_T(t) = P(T \leq t)$, and $G_T(t) = P(T > t) = 1 - F_T(t)$. **(6)**

- Detection events in non-overlapping time intervals are probabilistically independent. So,

$$P\{N(t+\Delta t) = 0\} = P\{N(t) = 0, \ N(t+\Delta t) - N(t) = 0\}$$
$$= P\{N(t) = 0\} \cdot P\{N(t+\Delta t) - N(t) = 0\} \quad \textbf{(7)}$$

where, $N(t)$ be the random number of event occur during $(0, t]$.

## 2. Derivation of $G_T(t)$, $F_T(t)$ and $E(T)$

Since the events of detection (and non-detection) are independent in non-overlapping time intervals,

$$G_T(t+\Delta t) = \underbrace{G_T(t)}_{A} \underbrace{(1 - [\gamma(t)\Delta t + o(\Delta t)])}_{B}. \quad \textbf{(8)}$$

Where:

- $A = P$ (non-detection from time to 0 to time $t$)

- $B = P$ (non-detection from time $t$ to time $t+\Delta t$).

Re-arranging terms,

$$\frac{G_T(t+\Delta t) - G_T(t)}{\Delta t} = -G_T(t)\gamma(t) - \frac{G_T(t)o(\Delta t)}{\Delta t}$$    . Letting     $\Delta t$    go     to

0 , $\frac{d}{dt}G_T(t) = -\gamma(t)G_T(t)$ .   The   solution   to   this   differential

equation is:

$$G_T(t) = \exp\left(-\int_0^t \gamma(s)ds\right) .$$    **(9)**

Therefore:

$$F_T(t) = 1 - G_T(t) = 1 - \exp\left(-\int_0^t \gamma(s)ds\right)$$

$$f_T(t) = \frac{d}{dt}F_T(t) = \gamma(t)\exp\left(-\int_0^t \gamma(s)ds\right)$$    .    **(10)**

$$E(T) = \int_{t=0}^\infty t\, f_T(t)dt = \int_{t=0}^\infty t\,\gamma(t)\exp\left(-\int_0^t \gamma(s)ds\right)dt$$

If    $\int_{t=0}^\infty f_T(t)dt = 1$  ,    then    $T$   is    a    proper    random    variable,

eventually detection is certain, and a slightly simpler

expression is possible. For any proper non-negative random

variable $T$:

$$E(T) = \int_{t=0}^\infty (1 - F_T(t))dt = \int_{t=0}^\infty \exp\left(-\int_0^t \gamma(s)ds\right)dt$$    .    **(11)**

The proof is sketched below in Figure 2.

$$\textcircled{A} = \int_x (1 - F(x))dx$$
$$= \text{area above } F(x) \text{ and below } 1$$
$$\approx \sum_{x_i} x_i \Delta F(x_i)$$
$$\approx \sum_{x_i} x_i f(x_i) \Delta x$$
$$\rightarrow \int_x x f(x)dx = E(X)$$
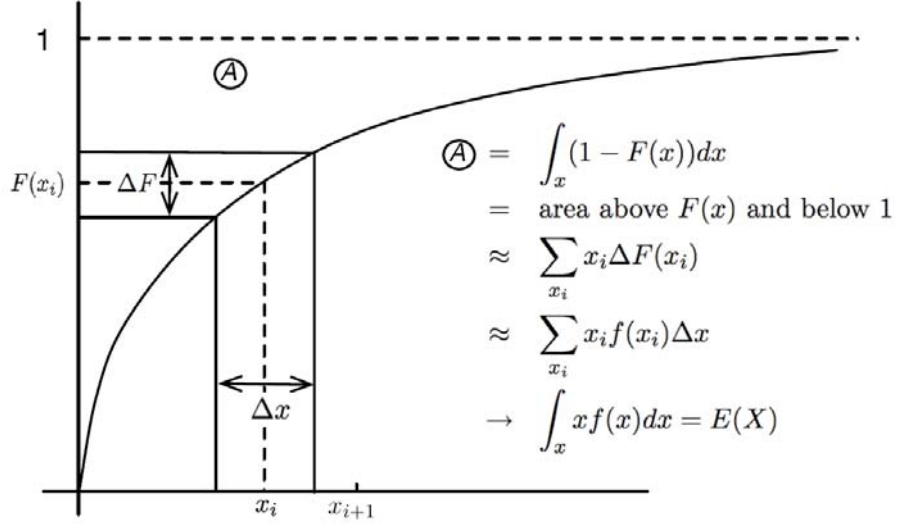
Figure 2.    $E(X)$ for Proper, Non-Negative Random Variables X

## C.    RANDOM SEARCH

### 1.    Characteristics of Random Search

Assume a search where:

a.    Searcher has a cookie-cutter sensor.

b.    Each small segment of the searcher's track is randomly and uniformly distributed over the search area.

c.    No search effort falls outside the search area.

d.    Target position is fixed in search area.

During any time interval $\Delta t$ , the searched area is $2RV\Delta t$ . Since this area is uniformly distributed over the search area $A$ , the probability of its covering the target is:

$$\frac{2RV\Delta t}{A} = P(\text{detection during } [t,\ t+\Delta t]) \ .$$

Thus, we have a constant detection rate with $\gamma = (2RV)/A$. Therefore:

$$F_T(t) = 1 - \exp(-2RVt/A)$$
$$f_T(t) = (2RV/A)\exp(-2RVt/A) \text{.} \tag{12}$$
$$E(T) = A/(2RV)$$

## 2.  Dynamic Enhancement of Random Search

Now assume the target with speed $U$ and the searcher with speed $V$ move randomly and independently over area $A$. The searcher sweep width is $2R$, and $T$ is the random time of initial detection. As before, we wish to compute $F_T(t) = P(T \leq t)$, and for $U << V$, we would expect that $F_T(t) \approx 1 - e^{-2RVt/A}$. And more generally, we will look for a speed $\tilde{V} > \max(V,U)$ such that $F_T(t) \approx 1 - e^{-2R\tilde{V}t/A}$.
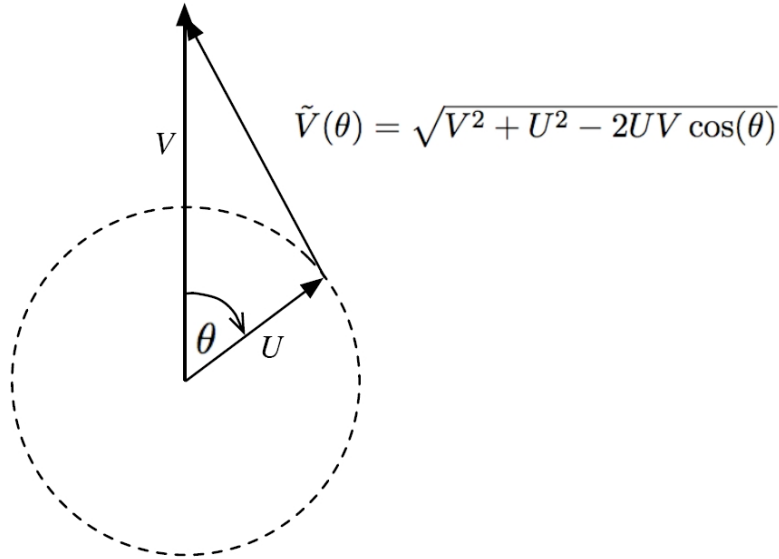


Figure 3.    Dynamic Enhancement of Random Search

Referring to Figure 3, $\tilde{V}(\theta)$ is the relative speed between searcher and target when $\theta$ is the difference in their courses. By the Law of Cosines, $\tilde{V}(\theta) = \sqrt{V^2 + U^2 - 2UV\cos(\theta)}$. We now assume that $\theta$ is uniformly distributed between 0 and $2\pi$, so the average $\tilde{V}(\theta)$ is:

$$
\begin{aligned}
\tilde{V} &= \int_{\theta=0}^{2\pi} V(\theta) f(\theta) d\theta \\
&= \int_{\theta=0}^{2\pi} \sqrt{V^2 + U^2 - 2UV\cos(\theta)} (\frac{1}{2\pi}) d\theta \\
&= \frac{1}{\pi} \int_{\theta=0}^{\pi} \sqrt{V^2 + U^2 - 2UV\cos(\theta)} d\theta \\
&\geq \max(V, U).
\end{aligned}
\tag{13}
$$

This equation suggests that in Random Search with dynamic enhancement, the searcher speed is effectively "enhanced" by the target speed. The author will confirm this by MATLAB simulation in Chapter III.

9

## 3. Random Search when Searcher and Target Patrol Areas are not Identical



**As** : Searcher's Search Area

**Ast** : Overlap Area

**At** : Target Movement Area
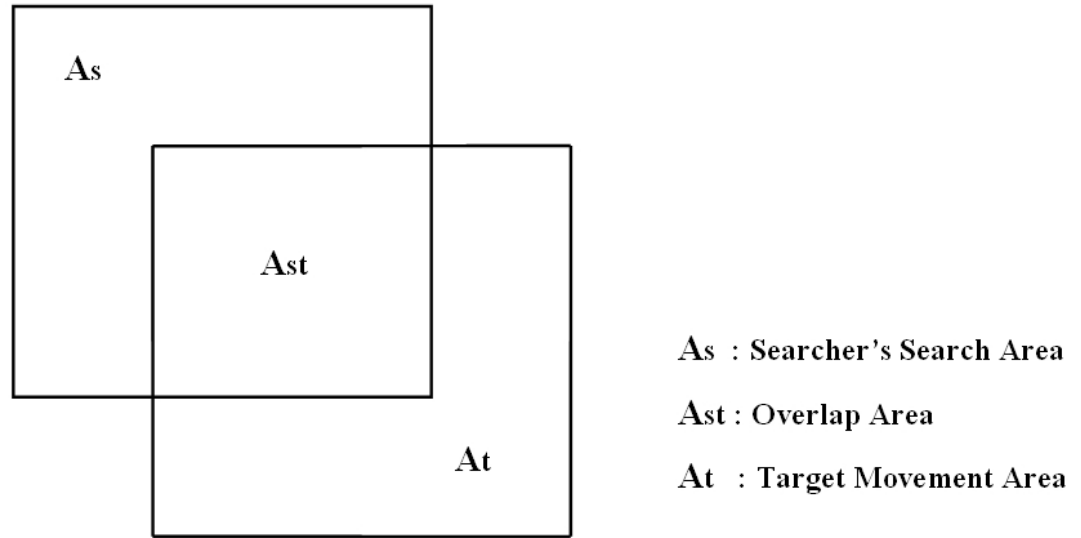
Figure 4.        Illustration of 'Search Area is not Completely Overlapped with Target Area'
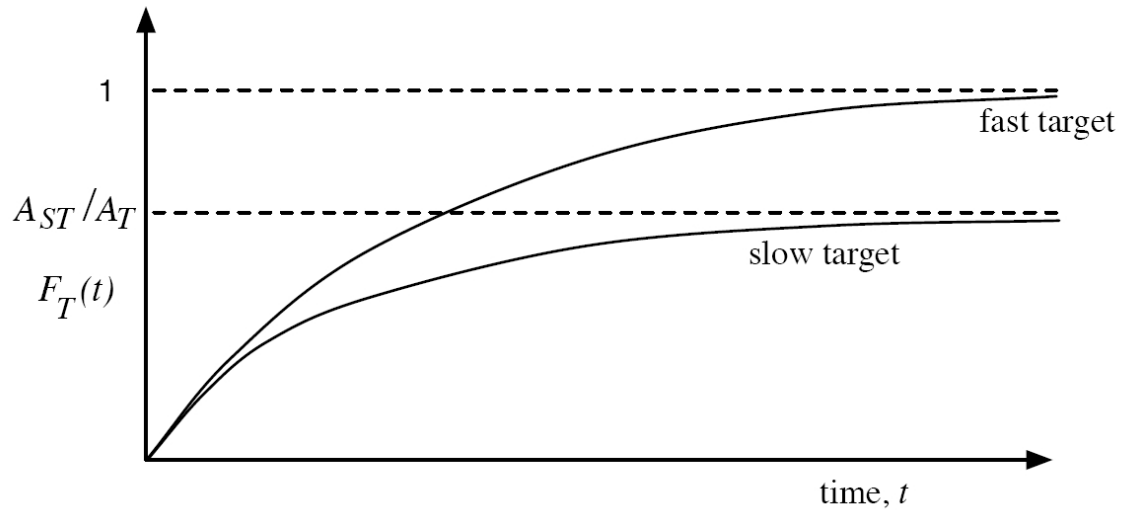


Figure 5.        $F_T(t)$ for Fast and Slow Target Assumption

As shown above in Figure 4, we now consider the situation where the search area is not completely overlapped with the target area. Two types of target behavior were assumed in order to get a simplified mathematical expression.

### a.   Slow Target

Assumptions:

- Target initial position is uniformly distributed over $A_t$.

- A target starting inside (outside) $A_s$ will remain so for the entire search time.

In this case $F_T(t)$ is:

$$F_T(t) = P(\text{detection by time t | target start in } A_{st})P(\text{target starts in } A_{st})+$$
$$P(\text{detection by time t | target starts outside } A_{st})P(\text{target starts outside in } A_{st})$$
$$= P(\text{detection by time t | target starts in } A_{st})P(\text{target starts in } A_{st}) \qquad \textbf{(14)}$$
$$= (\frac{A_{st}}{A_t})(1-\exp(-2R\tilde{V}t/A_s)).$$

### b.   Fast Target

Assumption:

- Target spends $(A_{st}/A_t)$ of the search time inside $A_s$.

By time $t$, this target has been available for detection for $(A_{st}/A_t)t$ time units. Therefore,

$$F_T(t) = 1-\exp(-2R\tilde{V}(A_{st}/A_t)t/A_s). \qquad \textbf{(15)}$$

11

In light of these assumptions, eventual detection for the fast target is assured. In contrast, the upper bound of $F_T(t)$ for the slow target is $(A_{st}/A_t)$. In this thesis, we will attempt to generalize the fast and slow target models to allow $F_T(t)$ to be estimated for intermediate target speeds.

# III. SIMULATION OF RANDOM SEARCH

## A. DESCRIPTION OF RANDOM SEARCH MODEL

### 1. Characteristics of the Searcher

The searcher is assumed to have a "cookie-cutter" sensor of radius $R$. In addition, the searcher is not allowed to search for a target outside the search area. Therefore the searcher's allowable position is limited by sensor radius $R$. For example, if the search region's $X$-axis length is 50nm and the $Y$-axis length is also 50nm, then the overall search area ($A_s$) is $50 \times 50 \text{nm}^2$. But the searcher's actual moving area ($A'$) is $(50-R) \times (50-R)\text{nm}^2$.

The searcher's initial position is uniformly distributed inside of the search region $A'$. After that, the searcher chooses his course randomly, independent of the target's movement. The course change event is determined by Poisson process with rate $\lambda$. It is assumed that the speed of the searcher is always faster than that of the target. This is allowed because the searcher and target roles can be reversed in the simulation, resulting in the same probability of detection.

### 2. Characteristics of the Target

The target's initial position is uniformly distributed over the search region $A$. The logic of the target movement is the same as that of the searcher, and the target has its own, independent Poisson process with course change rate $\lambda_t$.

**B.      COMPUTER ALGORITHM**

**1.      Inputs [Units]**

- Number of simulation replications, $N_{reps} = 500$.

- Maximum simulation time, $t_{max} = 150$ [hour].

- The length of search area of $X$ direction, $l_x = 150$ [nm].

- The length of search area of $Y$ direction, $l_y = 150$ [nm].

- Searcher speed, $V = 200$ [nm/hour].

- Target speed, $U$ [nm/hour].

- Searcher's detection range, $R = 2$ [nm].

- Searcher's course change rate, $\lambda_s$ [1/hour].

- Target's course change rate, $\lambda_t$ [1/hour].

- The unit time of simulation, $\Delta t$ [hour].

- The size of Search area, $A_s = 150^2$ [$nm^2$].

**2.      Functioning of the Program**

When a new replication begins, the initial positions of the searcher and the target are chosen from a Uniform Distribution over the search area. The only difference between position selection logic of the searcher and the target is caused by the searcher's sensor range ($R$). In order to prevent over-searching, the searcher's initial position should be limited to search area ($A'$).

The initial course is also drawn from a Uniform Distribution between 0 and $2\pi$. The subsequent course changes for searcher and target occur according to Poisson

Processes with rate $\lambda_s$ and $\lambda_t$ respectively. In particular, at each time step $\Delta t$, if $Uniform\_Random(0,1) < \lambda_s \Delta t$, then a new random course $C_s = Uniform\_Random(0,1) \times 2\pi$ is selected for the searcher. The course changes for the target are determined in the same manner by using $\lambda_t$.

When the searcher or target encounters an area boundary, then a random reflection occurs. After the reflection, the new course in radians is $Uniform\_Random(C_\perp - .5,\ C_\perp + .5)$, where $C_\perp$ is the perpendicular course from the reflection boundary. The parameter .5 was determined experimentally to prevent both "corner capture" and too many near "perpendicular reflections." (see Figure 6)
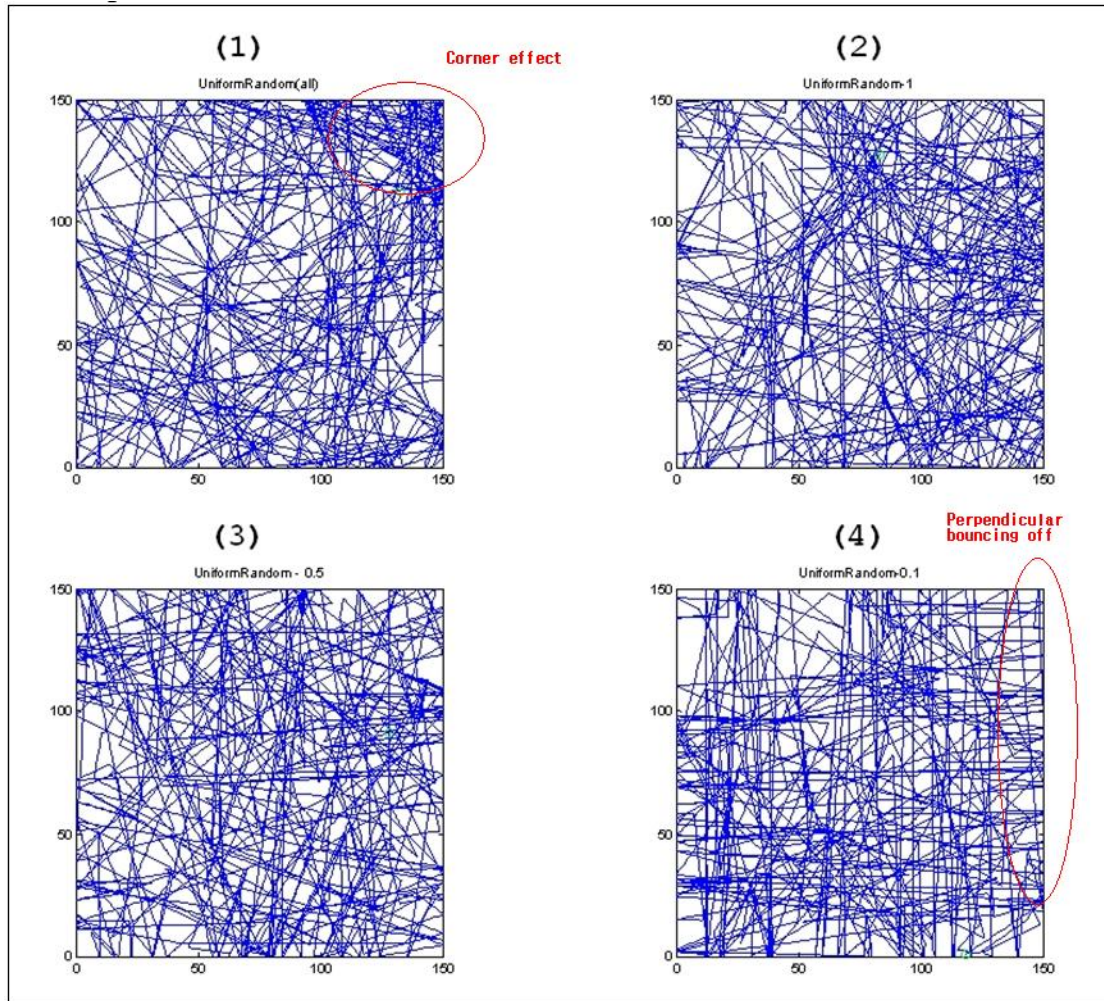
Figure 6.        Random Movement Behavior at the Edge
Depending on Different Parameters

At each time step, we store the position of the searcher and the target, and then measure the distance between them.

In order to closely approximate a continuous simulation, $\Delta t$ should be small. However, too small a $\Delta t$ means too many calculations, which in turn require an excessive time to simulate. How small a value of $\Delta t$ is sufficient to produce an accurate result?

16
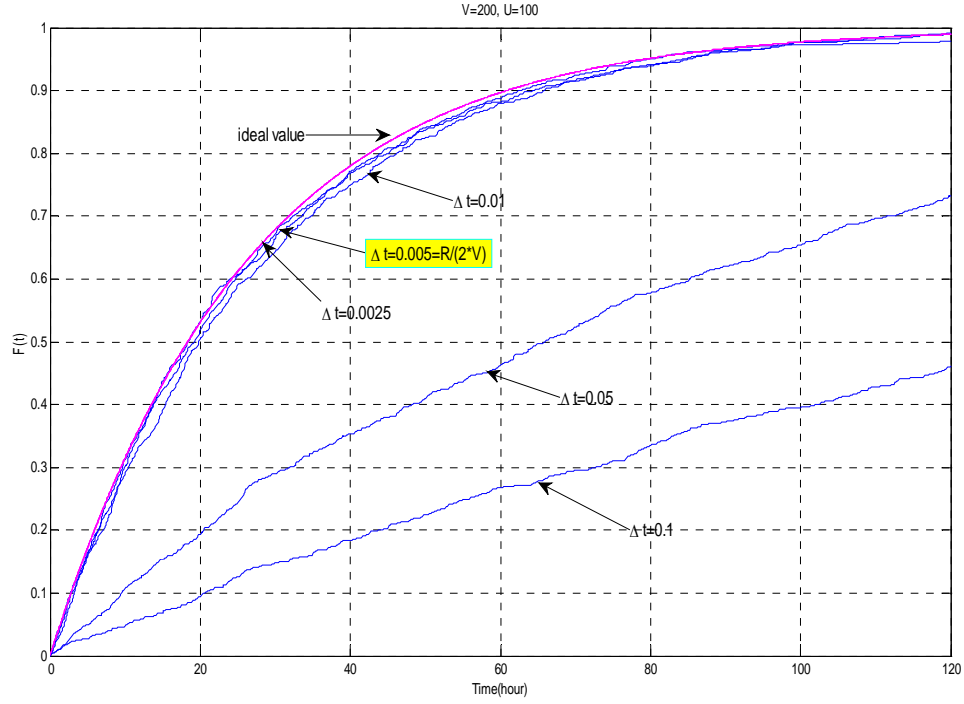
Figure 7.     $F_T(t)$ for Various Values of $\Delta t$

Figure 7 above shows that $\Delta t \approx \dfrac{R}{2V}$ is an appropriate value to use. Thus the searcher moves half the distance of the cookie-cutter sensor's radius at each time step.

The author also experimented with the Poisson course change rate $\lambda$ to produce searcher and target motion most closely satisfying the Random Search assumptions.

17
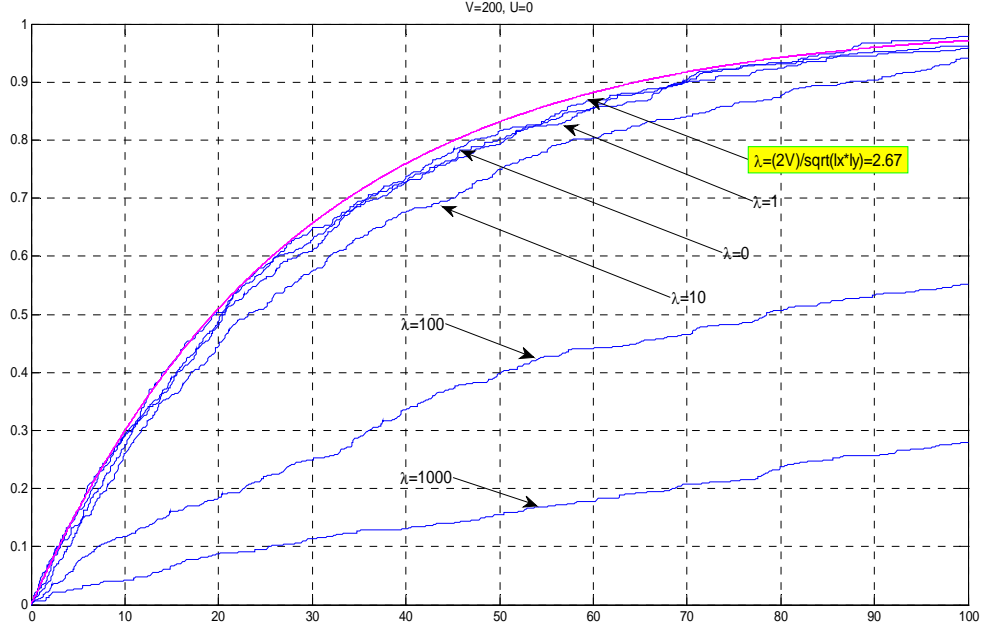
Figure 8.           $F_T(t)$ for Various Values of $\lambda$

As indicated in Figure 8, a good value is $\lambda = \dfrac{2V}{\sqrt{l_x \times l_y}}$ .

This formula implies that on the average, two course change events occur during the time required for the mover to go from edge to edge. If $\lambda$ is too small, the mover has a very small chance to change course before bouncing off the edge. On the other hand, if $\lambda$ is too large then the mover's position will be potentially limited to a small part of the searcher and target area (see Figure 9).
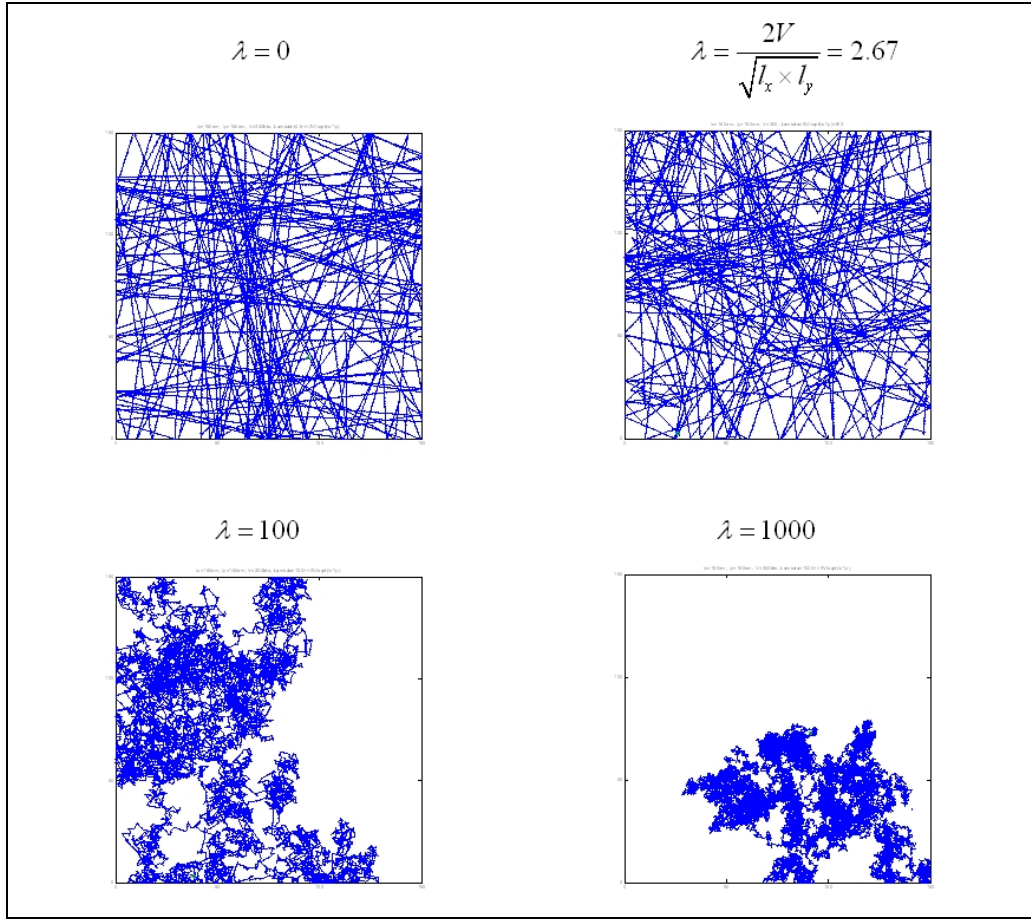
Figure 9.    Searcher's Movement Pattern for Various
Values of $\lambda$

## 3.    Output

Using the recommended values for the reflection parameter (0.5 radian) and course change rate ( $\lambda = \dfrac{2V}{\sqrt{l_x \times l_y}}$ ), the Random Search simulation produced results very close to the expected theoretical model.
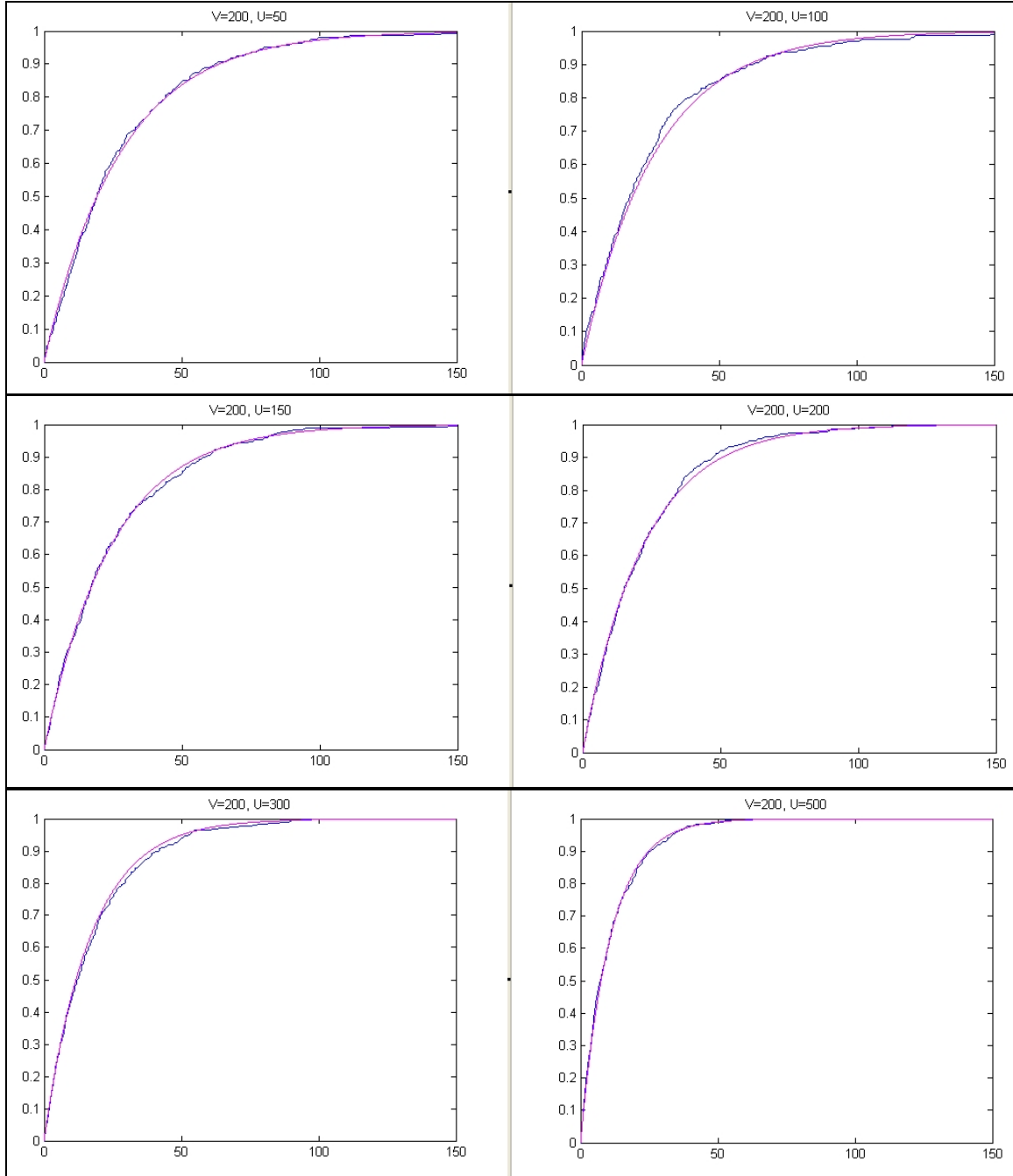
Figure 10.        $F_T(t)$ for Various Target Speeds

In Figure 10, the pink line is plots of following formula:

$$F_T(t) = 1 - (1 - \pi R^2 / A_s) \times \exp(-2R\tilde{V}t / A_s) \ . \tag{16}$$

The blue line represents simulation results. The results indicate that this MATLAB simulation is a valid representation of Random Search.

It is also clear that the movement of the target increases the opportunity for detection. Thus, if the searcher conducts a Random Search, the best strategy for the target is to be stationary.

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. SIMULATION OF THE EXTENDED RANDOM SEARCH MODEL WHERE SEARCHER AND TARGET AREAS ARE NOT COINCIDENT

## A. DESCRIPTION OF THE GENERALIZED RANDOM SEARCH MODEL

### 1. Characteristics of the Searcher and the Target

The only difference from the previous model is that the search area is now not completely overlapped with the target area.

## B. COMPUTER ALGORITHM

### 1. Inputs [Units]

- Number of simulation replications, $N_{reps} = 500$.

- Maximum simulation time, $t_{max} = 150$ [hour].

- The length of search area in $X$ direction, $l_x = 150$ [nm].

- The length of search area in $Y$ direction, $l_y = 150$ [nm].

- The length of target area in $X$ direction, $l_{xt} = 150$

- [nm].

- The length of target area in $Y$ direction, $l_{yt} = 150$ [nm].

- Searcher speed, $V = 200$ [nm/hour].

- Target speed, $U$ [nm/hour].

- Searcher's detection range, $R = 2$ [nm].

- Searcher's course change rate, $\lambda_s$ [1/hour].

- Target's course change rate, $\lambda_t$ [1/hour].

- The unit time of simulation, $\Delta t = \dfrac{R}{2V} = 0.005$ [hour].

- The size of search area, $A_s = 150^2$ [nm$^2$].

- The size of target area, $A_t = 150^2$ [nm$^2$].

- The size of overlap area, $A_{st} = 100^2$ [nm$^2$].

## 2.    Functioning of the Program

When this model was implemented in MATLAB, the faster mover's $\Delta t$ was used. For example, if the searcher's speed is 200kts and the target's speed is 100kts then, this simulation's $\Delta t$ is $\dfrac{2\text{nm}}{2 \times 200\text{kts}} = 0.005 \text{ hour}$.
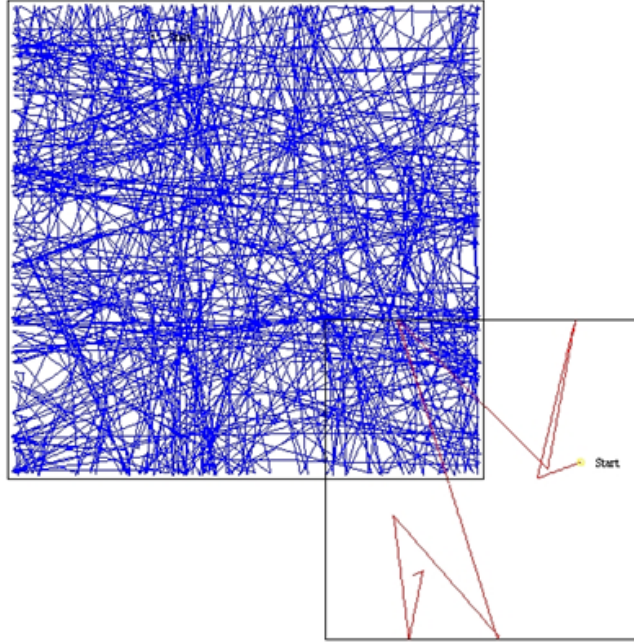


Figure 11.        Target Movement Behavior for $\lambda_t$ (V=200kts, U=2kts)

The image in Figure 11 results when using the following parameters: $t_{\max} = 200$ hours, search area $A_s = 150^2 \text{nm}^2$, target

24

moving area $A_t = 100^2 \text{nm}^2$, overlap area $A_{st} = 50^2 \text{nm}^2$, $V = 200\text{kts}$,

$U = 2\text{kts}$, $R = 2\text{nm}$, $\Delta t = \dfrac{2\text{nm}}{2 \times 200\text{kts}} = 0.005$ hours, $\lambda_s = \dfrac{2 \times 200\text{kts}}{\sqrt{150\text{nm} \times 150\text{nm}}} = 2.67/\text{hour}$

, and $\lambda_t = \dfrac{2 \times 2\text{kts}}{\sqrt{100\text{nm} \times 100\text{nm}}} = 0.04/\text{hour}$. For both the fast searcher and the slow target, there is an average of two course changes during the time the mover crosses its area.

### 3. Output

Models for the fast and slow target motion were introduced in Chapter II. Figure 12 shows simulation results for several target speeds plotted against the fast and slow target models:

1. Green line : fast target assumption

$$F_{Tfast}(t) = 1 - \exp(-2R\tilde{V}(A_{st}/A_t)t/A_s) \tag{17}$$

2. Pink line : slow target assumption

$$F_{Tslow}(t) = (A_{st}/A_t)(1 - \exp(-2R\tilde{V}t/A_s)) \tag{18}$$

3. Blue line : simulation results
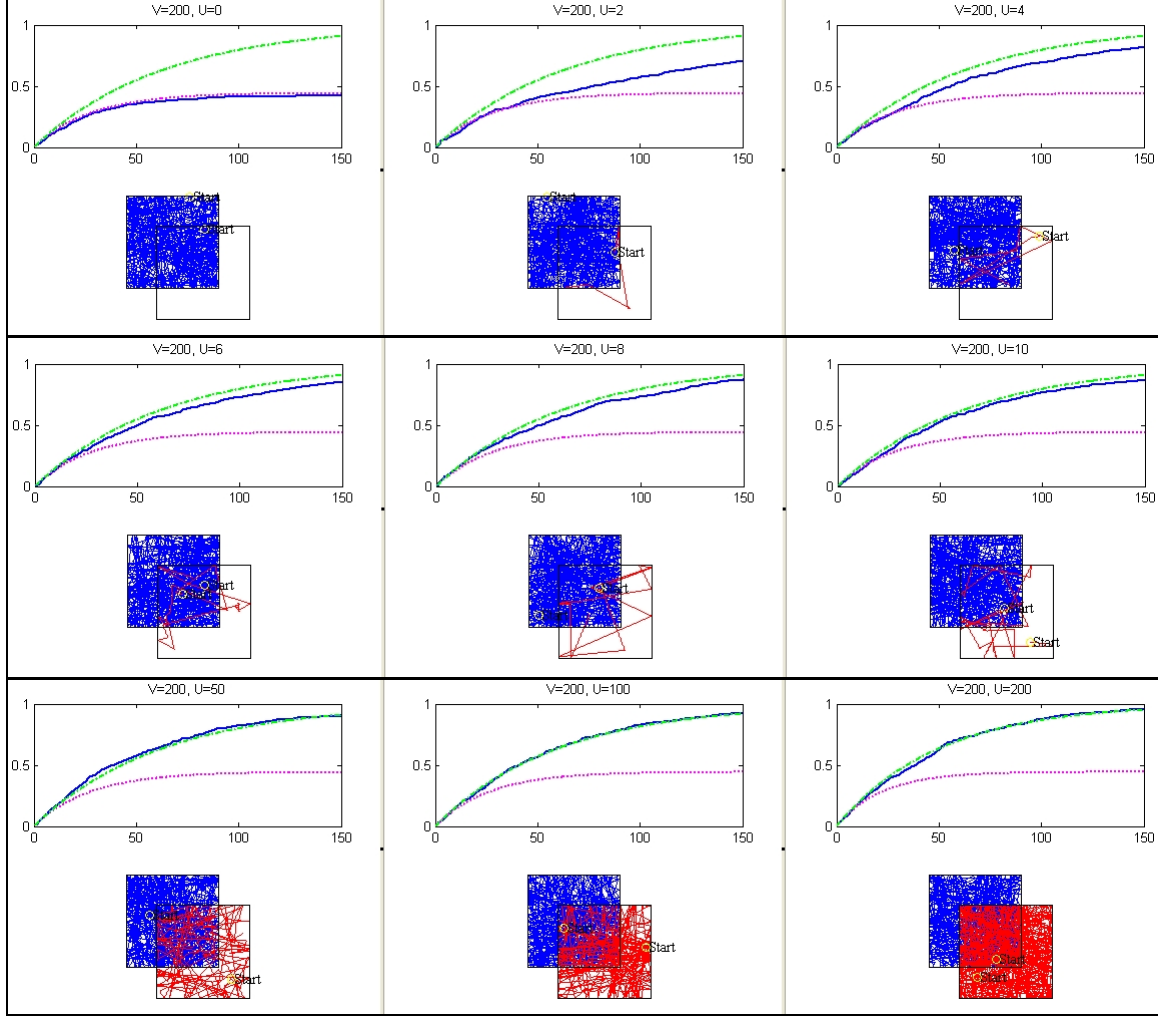
Figure 12.        $F_T(t)$ for Various Target Speed, Overlap Area
$$A_{st} = 100nm \times 100nm$$

The fast target and slow target equations can be combined as follows to approximate the simulation results for any target speeds:

$$F_{Tcombined}(t) = (\underbrace{\exp(-\xi\sqrt{t})}_{\alpha})F_{Tslow}(t) + (\underbrace{1-\exp(-\xi\sqrt{t})}_{1-\alpha})F_{Tfast}(t)$$

**(19)**

where $\xi = \dfrac{A_{st}U}{\sqrt{A_s A_t RV}}$  [ $1/\sqrt{\text{time}}$ ] .

26

$\alpha$ decreases from 1 to 0 as the target speed $U$ increases and as time goes to infinity, resulting in the fast target equation (17). Figure 13 shows plots of simulation results and $F_{Tcombined}(t)$. The red line in the Figure 13 represents the results of equation (19).
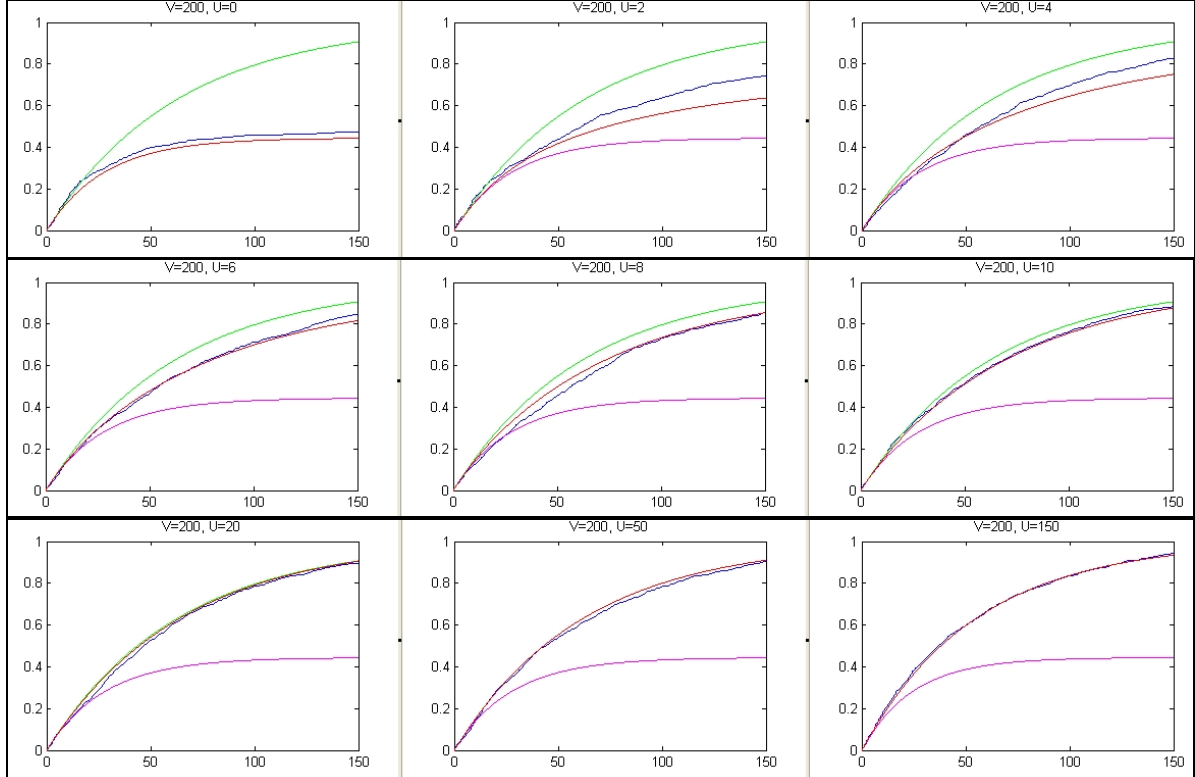


Figure 13.    The Comparison between Simulation Results and $F_{Tcombined}(t)$

Given these results, it appears that equation (19) can be used as a conservative estimate (that is, a lower bound) of $F_T(t)$ when searcher and target patrol areas are not identical.

THIS PAGE INTENTIONALLY LEFT BLANK

# V. SIMULATION OF A SYSTEMATIC RANDOM SEARCH AGAINST A RANDOMLY MOVING TARGET

## A. DESCRIPTION OF 'SYSTEMATIC SEARCH' MODEL

Actual searchers cannot perform Exhaustive Search, but it is often approximated by parallel sweeps, which is similar to mowing the lawn. This is called 'Systematic Search'. Other forms of Systematic Search include spiral-in and spiral-out tracks.



(1)                              (2)
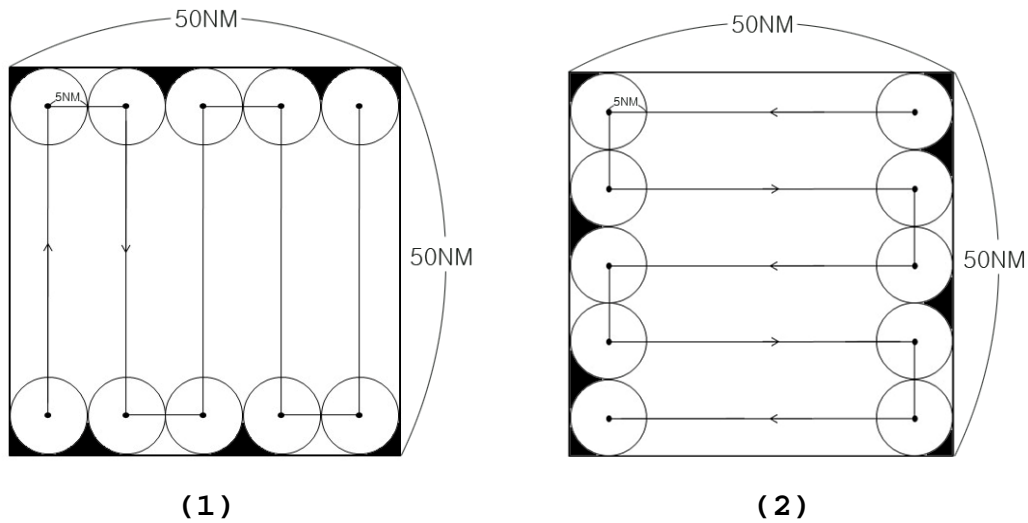
Figure 14.         Simple Examples of 'Systematic Search' Patterns

Figure 14 illustrates simplified examples of a 'Systematic Search' track. We examine the two types of going and returning 'Systematic Search' patterns. One is a 'Reverse Course Systematic Search' in which the searcher goes back via the exact same track to the starting point. The other is a 'Cross Course Systematic Search' in which the searcher follows the track shown in Figure 14(1) and then goes back via the Figure 14(2) track.

## B.    COMPUTER ALGORITHM

### 1.    Input [Units]

- Number of simulation replications, $N_{reps} = 1000$.

- Maximum simulation time, $t_{max} = 54.72$ [hour].

- The length of search area of $X$ direction, $l_x = 148$ [nm].

- The length of search area of $Y$ direction, $l_y = 148$ [nm].

- Searcher speed, $V = 200$ [nm/hour].

- Target speed, $U$ [nm/hour].

- Searcher's detection range, $R = 2$ [nm].

- Target course change rate, $\lambda_t = \dfrac{2U}{\sqrt{l_x \times l_y}}$ [1/hour].

- The unit time of simulation, $\Delta t = \dfrac{R}{2V} = 0.005$ [hour].

- The searcher's moving distance during each time step, $d_{unit\_time} = V \times \Delta t = 1$ [nm].

- The size of search area, $A_s = 148^2$ [$nm^2$].

### 2.    Comparison of the Searcher's Movement Patterns

Simulation results, illustrated in Figure 15, indicate very little difference between the two search paths. The author postulates that any reasonable Systematic Search plan (e.g. parallel sweeps, spiral-in, or spiral-out) will be effective as long as the searcher attempts to cover all points in the search area with the sensor one time before any point is covered twice.
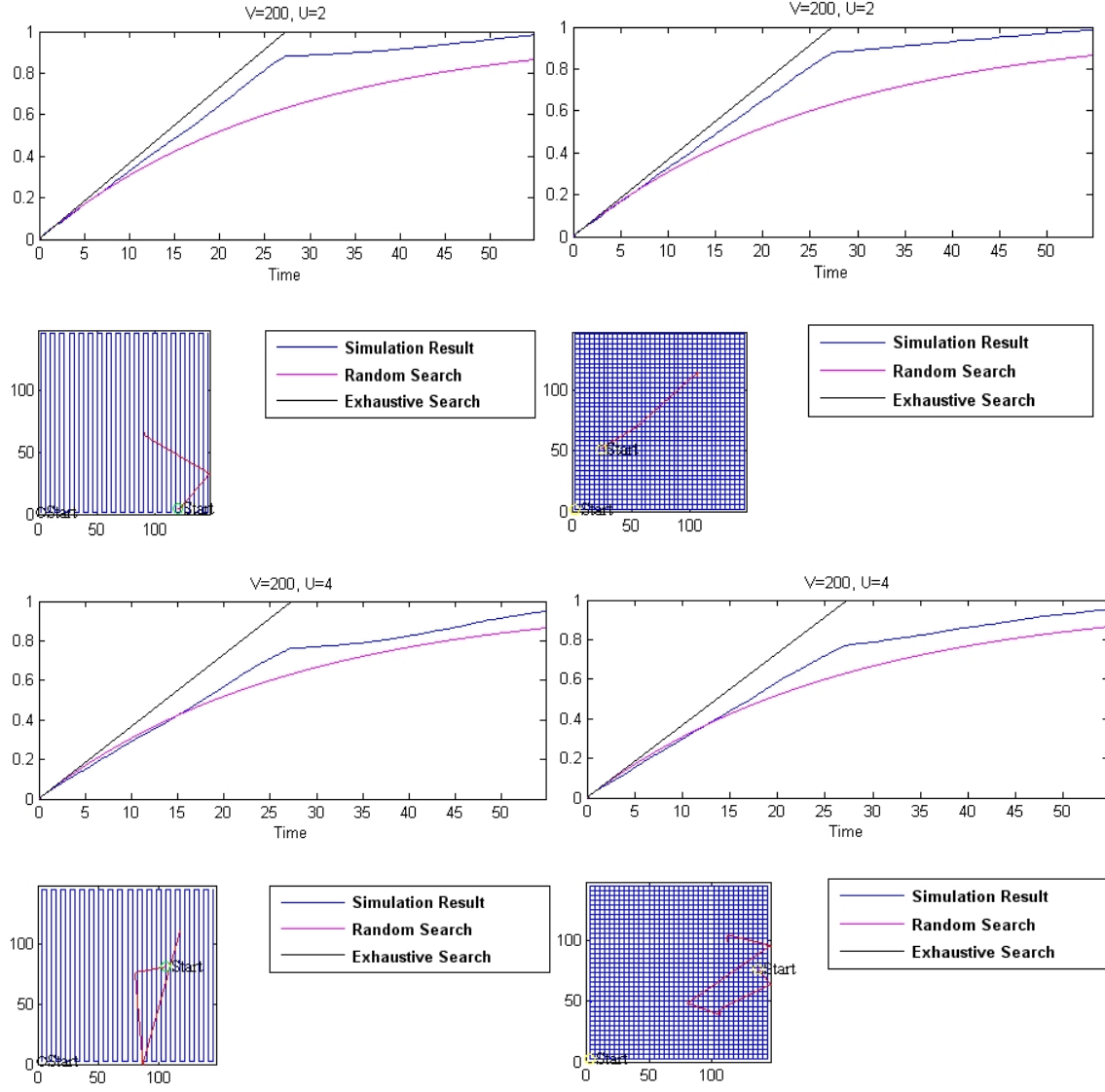
Figure 15.    Comparison Between Reverse and Cross Course


In Figure 15, each line is defined as follows:

Exhaustive Search (Black line): $F_T(t) = (2\tilde{V}Rt)/A_s$.

Random Search (Pink line): $F_T(t) = 1 - (1 - \pi R^2/A_s) \times \exp(-2R\tilde{V}t/A_s)$.

Simulation Results(Blue line).

As expected, there is very little difference between the reverse and the cross course's simulation results.

## 3. Output
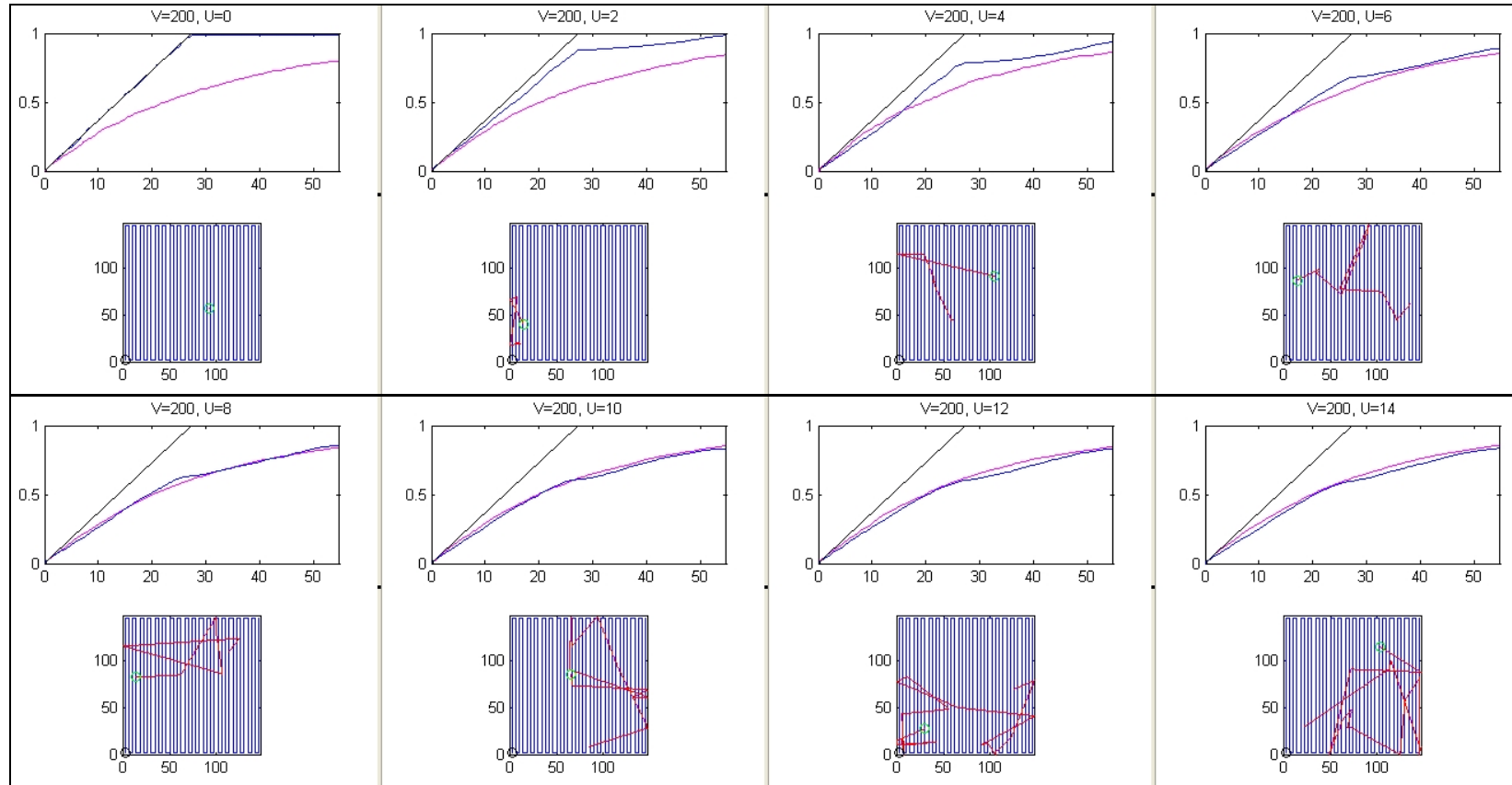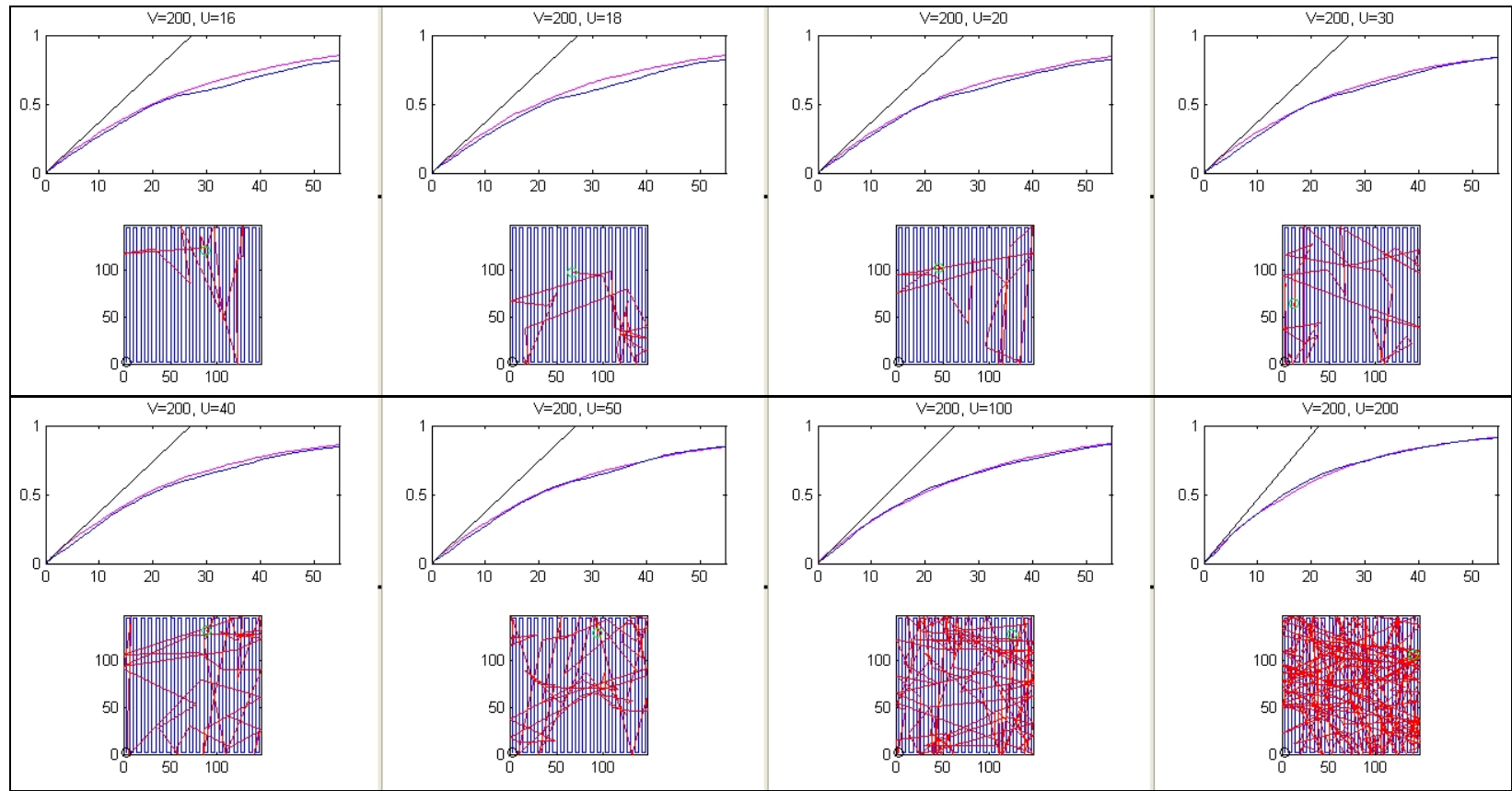
Figure 16.        $F_T(t)$ for Various Target Speed (2 Times Sweep)

In general, it is thought that a simulation of Systematic Search always performs better than Random Search. Figure 16 suggests that this is true.

The measure of effectiveness used when evaluating search plans was "Incomplete Mean Time to Detection evaluated at time $\bar{t}$ ( $\text{IMTD}(\bar{t})$ )," defined as, $\text{IMTD}(\bar{t}) = \int_0^{\bar{t}} (1 - F_T(x))dx$. $\text{IMTD}(\bar{t})$ is used as a surrogate for Mean time to detection( $E(T)$ ). In fact, $\lim_{\bar{t} \to \infty} \text{IMTD}(\bar{t}) = E(T)$. We use $\text{IMTD}(\bar{t})$ because we often cannot run the search simulation long enough to accurately estimate $E(T)$ (see Figure 17).



V=200, U=20

Area 1 + Area 2 $= \int_{x \geq 0} (1 - F_T(x))dx = E(T)$ = Mean time to detection

Area 1 $= \int_0^{50} (1 - F_T(x))dx$ = Incomplete mean time to detection

$\therefore$ Incomplete mean time to detection $<$ Mean time to detection .
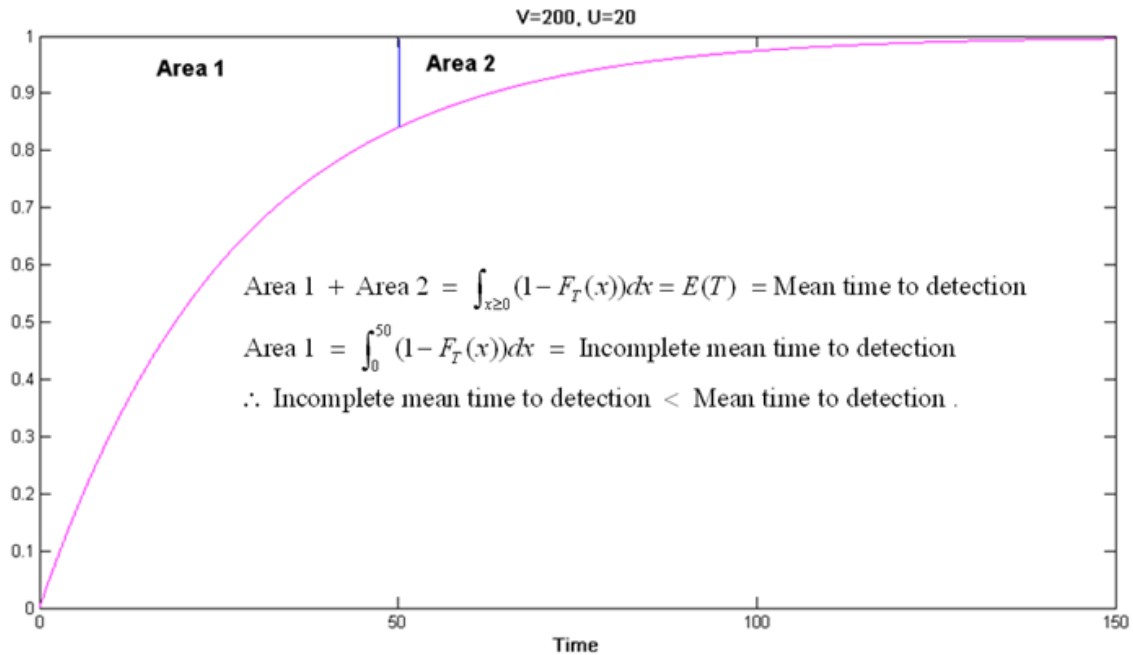
Figure 17.    Example of 'Incomplete Mean Time to Detection' where $\bar{t}$ =50 hours.

We assume that the target would like to select a speed maximizing $\mathrm{IMTD}(\overline{t})$. In Figure 18, $\mathrm{IMTD}(\overline{t})$ is dramatically increased when the target speed changed from 0kts to 8kts. And there are no significant changes between 8kts and 44kts. For target speeds greater than about 44kts, $\mathrm{IMTD}(\overline{t})$ steadily decreases.
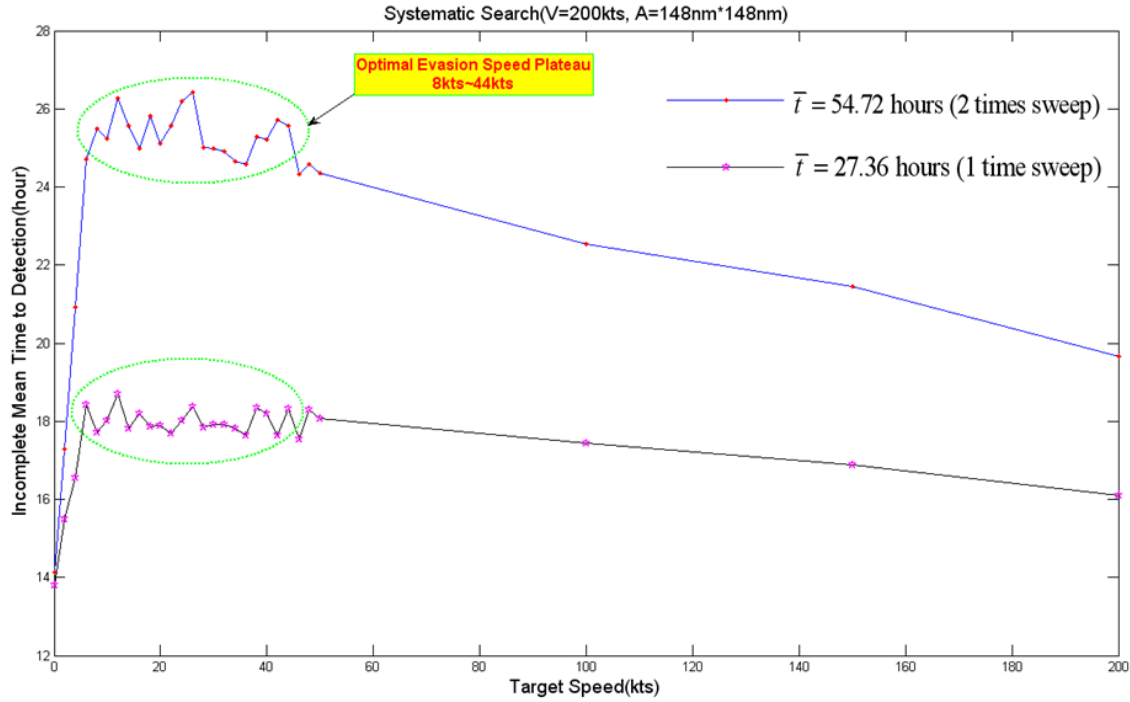


Figure 18.     Incomplete Mean Time to Detection for Various $\overline{t}$

Naval Postgraduate School Professor, Alan R. Washburn, suggested this when he wrote, "a target that wishes to avoid detection might actually choose to move around at $U = .2V$, on the grounds that this is enough motion to prevent an Exhaustive Search, but nonetheless increases the equivalent searcher speed by only 1%."[2] After that speed, as Professor Washburn expected, the $\mathrm{IMTD}(\overline{t})$ decreases because the fast

---

[2] Alan R. Washburn, Search and detection, 4th ed. 6-3.

random movement of the target runs it into the searcher more often than away from the searcher. Figure 18 shows that even though the speed of target increases from 0kts to 40kts, which is a speed enough to significantly degrade an Exhaustive Search, the increase of $\tilde{V}$ is only 1% of searcher speed.

All things considered, the author suggests that the optimal evasion speed for target evading a Systematic Search is approximately 0.05 to 0.2 of the searcher speed.
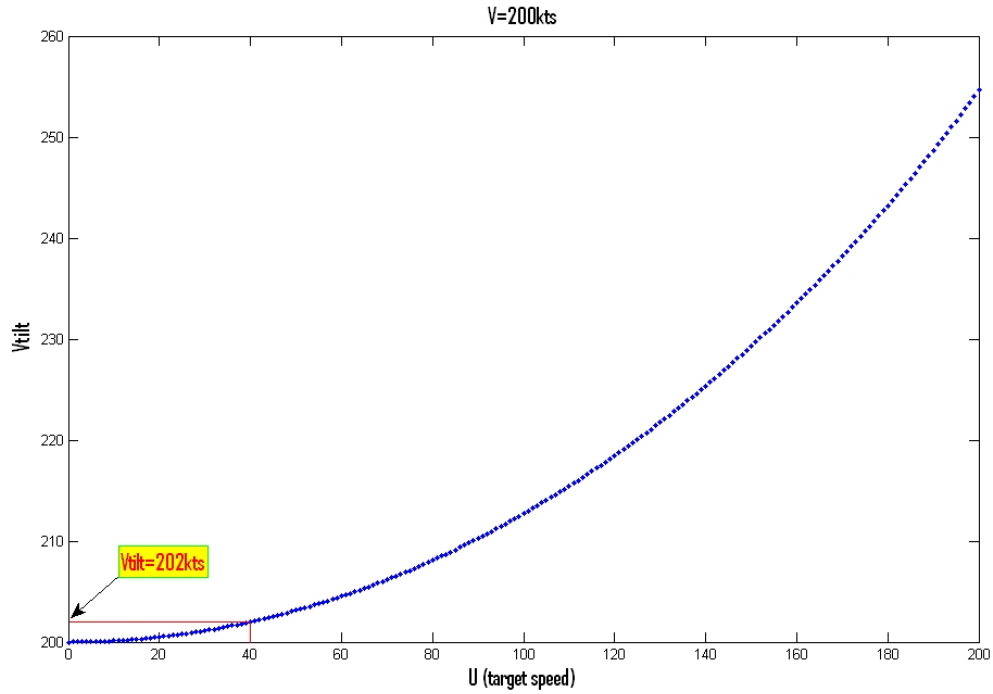


Figure 19.       The Relationship Between Target Speed($U$) and $\tilde{V}$

# VI. CONCLUSIONS

## A. CONCLUSIONS AND RECOMMENDATIONS

The main contributions of this thesis can be narrowed down to two results. The first result is a generalization of the Random Search formula which allows both searcher and target to move and does not require the patrol areas to be identical.

The other result is that the best speed for a randomly moving target evading a Systematic Search ranges from 0.05 to 0.2 of the searcher speed.

In this thesis, the searcher always uses a cookie-cutter sensor. We could develop a potentially more realistic model by relaxing the cookie-cutter sensor assumption in future studies. For instance, if we assume the probability of target detection with sensor range R is 0.8, then

$$P(\text{detection}) = P_d = \begin{cases} 1, & \text{if } Uniform\_Random(0,1) \leq 0.8 \ \& \ \text{range} \leq R \\ 0, & \text{otherwise .} \end{cases}$$

In addition, the comparison of results when using various kinds of sensor types such as cookie-cutter sensor, inverse-cube law sensor, and triangle sensor would be worthwhile to study.

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX

## A.    MATLAB CODE

### 1.    Random Search Model

```
Nreps=500;                %number of simulation replications
tmax=150;                 %max. simulation time (hr)
lx=150;                   %search area length in x direction   (nm)
ly=150;                   %search area length in y direction   (nm)
V=200;                    %searcher speed (nm/hr)
U=50;                     %target speed (nm/hr)
R=2;                      %searcher detection range (nm)
dt=R/(2*max(V,U));        %delta t (hours)
lams=(2*V)/sqrt(lx*ly);   %searcher course change rate (1/hr)
lamt=(2*U)/sqrt(lx*ly);   %target course change rate (1/hr)
Xs=zeros(1,tmax/dt+1);    %initialize x-position to zero(searcher)
Ys=Xs;                    %initialize y-position to zero(searcher)
Cs=Xs;                    %initialize searcher course to zero
Xt=Xs;                    %initialize x-position to zero(target)
Yt=Xs;                    %initialize y-position to zero(target)
Ct=Xs;                    %initialize target course to zero
CumDet=Xs;                %initialize cumulative detection stats
T=0:dt:tmax;              %simulation time vector
A=lx*ly;                  %search area
for n=1:Nreps             %main simulation loop
xs=rand*(lx-2*R)+R;       %initial searcher and target x and y positions
ys=rand*(ly-2*R)+R;
xt=rand*lx;
yt=rand*ly;
cs=rand*2*pi;             %initial searcher course
ct=rand*2*pi;             %initial target course
t=0;                      %set simulation time to 0
tindex=1;                 %initialize time index to 1
Xs(tindex)=xs;            %save initial searcher x position
Ys(tindex)=ys;            %save initial searcher y position
Cs(tindex)=cs;            %save initial searcher course
Xt(tindex)=xt;            %save initial target x position
Yt(tindex)=yt;            %save initial target y position
Ct(tindex)=ct;            %save initial target course
for t=1:tmax/dt           %inner loop
    tindex = tindex+1;    %update simulation time index
    if rand<lams*dt;      %Poisson course change rate.
    cs=rand*2*pi;          %New courses uniform (0,2*pi).
    end
    if Xs(tindex-1)<R; cs=(rand-.5);end
    if Xs(tindex-1)>(lx-R); cs=pi+(rand-.5);end
    if Ys(tindex-1)<R; cs=pi/2+(rand-.5);end
    if Ys(tindex-1)>(ly-R); cs=-pi/2+(rand-.5);end
    Xs(tindex) = Xs(tindex-1)+V*dt*cos(cs);    %Update x and y positions
    Ys(tindex) = Ys(tindex-1)+V*dt*sin(cs);
```

```matlab
        Cs(tindex)=cs;
        if rand<lamt*dt; ct=rand*2*pi;end
        if Xt(tindex-1)<0; ct=(rand-.5);end
        if Xt(tindex-1)>lx; ct=pi+(rand-.5);end
        if Yt(tindex-1)<0; ct=pi/2+(rand-.5);end
        if Yt(tindex-1)>ly; ct=-pi/2+(rand-.5);end
        Xt(tindex) = Xt(tindex-1)+U*dt*cos(ct);    %Update x and y positions
        Yt(tindex) = Yt(tindex-1)+U*dt*sin(ct);
        Ct(tindex)=ct;
        end                      %inner loop (time increasing from 0 to tmax)
        CumDet = CumDet + cummax((Xs-Xt).^2 + (Ys-Yt).^2 <= R^2);
        end                      %outer loop (simulation replications)
Probability=CumDet./Nreps;
RSfun=1-(1-pi*R*R/A)*exp(-2*R*vtilde(V,U)*T/A);
plot(T,CumDet/Nreps,'b-', T,RSfun,'m-'), axis([0,tmax,0,1])
```

## 2. 'Patrol Areas are not Identical' Model

```
Nreps=500;
tmax=150;
lx=150;                         %search area length in x direction (nm)
ly=150;                         %search area length in y direction (nm)
lx_t=150;                       %target area length in x direction (nm)
ly_t=150;                       %target area length in y direction (nm)
V=200;                          %searcher speed (nm/hr)
U=15;                           %target speed (nm/hr)
R=2;                            %searcher detection range (nm)
dt=R/(2*max(V,U));              %delta t (hours)
lams=(2*V)/sqrt(lx*ly);         %searcher course change rate (1/hr)
lamt=(2*U)/sqrt(lx_t*ly_t);     %searcher course change rate (1/hr)
A=lx*ly;                        %search area(As)
Ast=100*100;                    %overlap area
At=lx_t*ly_t;                   %target area(At)
Xs=zeros(1,tmax/dt+1);
Ys=Xs;
Cs=Xs;
Xt=Xs;
Yt=Xs;
Ct=Xs;
CumDet=Xs;
T=0:dt:tmax;
for n=1:Nreps
xs=rand*(lx-2*R)+R;
ys=rand*(ly-2*R)+R+50;
xt=rand*lx_t+50;
yt=rand*ly_t;
cs=rand*2*pi;
ct=rand*2*pi;
t=0;
tindex=1;
Xs(tindex)=xs;                  %save initial searcher x position
Ys(tindex)=ys;                  %save initial searcher y position
Cs(tindex)=cs;                  %save initial searcher course
Xt(tindex)=xt;                  %save initial target x position
Yt(tindex)=yt;                  %save initial target y position
Ct(tindex)=ct;                  %save initial target course
for t=1:tmax/dt
    tindex = tindex+1;
    if rand<lams*dt;
    cs=rand*2*pi;
    end
    if Xs(tindex-1)<R; cs=(rand-.5);end
    if Xs(tindex-1)>(lx-R); cs=pi+(rand-.5);end
    if Ys(tindex-1)<(R+50); cs=pi/2+(rand-.5);end
    if Ys(tindex-1)>(ly-R+50); cs=-pi/2+(rand-.5);end
    Xs(tindex) = Xs(tindex-1)+V*dt*cos(cs);
    Ys(tindex) = Ys(tindex-1)+V*dt*sin(cs);
    Cs(tindex)=cs;
    if rand<lamt*dt; ct=rand*2*pi;end
    if Xt(tindex-1)<50; ct=(rand-.5);end
    if Xt(tindex-1)>(lx_t+50); ct=pi+(rand-.5);end
```

```
    if Yt(tindex-1)<0; ct=pi/2+(rand-.5);end
    if Yt(tindex-1)>ly_t; ct=-pi/2+(rand-.5);end
    Xt(tindex) = Xt(tindex-1)+U*dt*cos(ct);
    Yt(tindex) = Yt(tindex-1)+U*dt*sin(ct);
    Ct(tindex)=ct;
    end
    CumDet = CumDet + cummax((Xs-Xt).^2 + (Ys-Yt).^2 <= R^2);
    if n/50 == floor(n/50), n, end;
end
RSfun_slow=(Ast/At)*(1-exp(-(2*R*vtilde(V,U)*T)/A));
RSfun_fast=1-exp(-(2*R*(Ast/At)*vtilde(V,U)*T)/A);
subplot(2,1,1)
plot(T,CumDet/Nreps,'b-', T,RSfun_slow,'m-',T,RSfun_fast,'g'),
axis([0,tmax,0,1])
subplot(2,1,2)
xline=[150 0 0 150 150 50 50 200 200 50 50];
yline=[50 50 200 200 50 50 150 150 0 0 50];
plot(Xs,Ys,'b-',Xt,Yt,'r
',Xs(1),Ys(1),'yo',Xt(1),Yt(1),'yo',xline,yline,'k'),axis off
hold on
axis([0 lx+50 0 ly+50]), axis square
scale=lx+50;
text(Xs(1)+scale/40,Ys(1),'Start','FontSize',10,'FontName','Times');
text(Xt(1)+scale/40,Yt(1),'Start','FontSize',10,'FontName','Times');
hold off
```

### 3.    Systematic Search Model

```matlab
Nreps=1000;                %number of simulation replications
lx=148;                    %search area length in x direction  (nm)
ly=148;                    %search area length in y direction  (nm)
V=200;                     %searcher speed (nm/hr)
U=150;                     %target speed (nm/hr)
R=2;                       %searcher detection range (nm)
dt=R/(2*V);                %delta t (hours)
unit_time_movement=V*dt;
tmax=dt*10944;             %max. simulation time (hr)=54.72 hours
lams=(2*V)/sqrt(lx*ly);    %searcher course change rate (1/hr)
lamt=(2*U)/sqrt(lx*ly);    %target course change rate (1/hr)
Xs=zeros(1,tmax/dt+1);     %initialize x-position to zero(searcher)
Ys=Xs;                     %initialize y-position to zero(searcher)
Cs=Xs;                     %initialize searcher course to zero
Xt=Xs;                     %initialize x-position to zero(target)
Yt=Xs;                     %initialize y-position to zero(target)
Ct=Xs;                     %initialize target course to zero
CumDet=Xs;                 %initialize cumulative detection stats
CumDet_s=Xs;               %initialize cumulative detection stats
T=0:dt:tmax;               %simulation time vector
A=lx*ly;                   %search area
%-----------------------define Searcher's coordinate
xs_s=R;                    %initial x coordinate(searcher)
for h=1:36
   xs_column=linspace(4*h-2,4*h-2,144); xs_s=[xs_s xs_column];
   xs_row=(4*h-1):unit_time_movement:((2*R)/unit_time_movement+4*h-2);
   xs_s=[xs_s xs_row];
end
   xs_f=linspace(146,146,144);
   xs_s=[xs_s xs_f];
for b=36:-1:1
   xs_column_b=linspace(4*b+2,4*b+2,144); xs_s=[xs_s xs_column_b];
   xs_row_b=(4*b+1):-unit_time_movement:(4*b+2-
   (2*R)/unit_time_movement);
   xs_s=[xs_s xs_row_b];
end
   xs_b=linspace(2,2,144);
   xs_s=[xs_s xs_b];
   ys_s=2:1:146;           %define target's coordinate
 for z=1:18
   ys_row=linspace(146,146,4); ys_s=[ys_s ys_row];
   ys_column=145:-1:2; ys_s=[ys_s ys_column];
   ys_row=linspace(2,2,4); ys_s=[ys_s ys_row];
   ys_column=3:1:146; ys_s=[ys_s ys_column];
 end
 for z=1:18
   ys_column=145:-1:2; ys_s=[ys_s ys_column];
   ys_row=linspace(2,2,4); ys_s=[ys_s ys_row];
   ys_column=3:1:146; ys_s=[ys_s ys_column];
   ys_row=linspace(146,146,4); ys_s=[ys_s ys_row];
 end
   ys_b=145:-1:2; ys_s=[ys_s ys_b];
%----------------------------------------
```

```
for n=1:Nreps              %main simulation loop
xs=rand*(lx-2*R)+R;        %initial searcher and target x and y positions
ys=rand*(ly-2*R)+R;
xt=rand*lx;
yt=rand*ly;
cs=rand*2*pi;              %initial searcher course
ct=rand*2*pi;              %initial target course
t=0;                       %set simulation time to 0
tindex=1;                  %initialize time index to 1
Xs(tindex)=xs;             %save initial searcher x position
Ys(tindex)=ys;             %save initial searcher y position
Cs(tindex)=cs;             %save initial searcher course
Xt(tindex)=xt;             %save initial target x position
Yt(tindex)=yt;             %save initial target y position
Ct(tindex)=ct;             %save initial target course
for t=1:tmax/dt            %inner loop
    tindex = tindex+1;     %update simulation time index
    if rand<lams*dt;       %Poisson course change rate.
    cs=rand*2*pi;          %New courses uniform (0,2*pi).
    end
    if Xs(tindex-1)<R; cs=(rand-.5);end
    if Xs(tindex-1)>(lx-R); cs=pi+(rand-.5);end
    if Ys(tindex-1)<R; cs=pi/2+(rand-.5);end
    if Ys(tindex-1)>(ly-R); cs=-pi/2+(rand-.5);end
    Xs(tindex) = Xs(tindex-1)+V*dt*cos(cs);
    Ys(tindex) = Ys(tindex-1)+V*dt*sin(cs);
    Cs(tindex)=cs;
    if rand<lamt*dt; ct=rand*2*pi;end
    if Xt(tindex-1)<0; ct=(rand-.5);end
    if Xt(tindex-1)>lx; ct=pi+(rand-.5);end
    if Yt(tindex-1)<0; ct=pi/2+(rand-.5);end
    if Yt(tindex-1)>ly; ct=-pi/2+(rand-.5);end
    Xt(tindex) = Xt(tindex-1)+U*dt*cos(ct);
    Yt(tindex) = Yt(tindex-1)+U*dt*sin(ct);
    Ct(tindex)=ct;
end                        %inner loop (time increasing from 0 to tmax)
CumDet = CumDet + cummax((Xs-Xt).^2 + (Ys-Yt).^2 <= R^2);
CumDet_s = CumDet_s + cummax((xs_s-Xt).^2 + (ys_s-Yt).^2 <= R^2);
end                        %outer loop (simulation replications)
Exhautive_Search=2*R*vtilde(V,U)*T/A;
subplot(2,1,1)
plot(T,CumDet/Nreps,'b-', T,CumDet_s/Nreps,'m-
',T,Exhautive_Search,'k'), axis([0,tmax,0,1])
subplot(2,1,2)
plot(xs_s,ys_s,Xt,Yt,'r-',xs_s(1),ys_s(1),'ko',Xt(1),Yt(1),'go'),axis([0
lx 0 ly]),axis square
```

# LIST OF REFERENCES

Chudnovsky, David V., and Gregory V. Chudnovsky, Search
theory: Some recent developments, (New York: Marcel
Dekker, 1989).

Gilat, Amos. MATLAB: An introduction with applications, 2nd
ed., (Hoboken, N.J.: John Wiley, 2005).

Iida, Koji. Studies on the optimal search plan, (Berlin, New
York: Springer-Verlag, 1992).

Koopman, Bernard Osgood. Search and screening, (New York:
Pergamon Press, 1980).

NATO Advanced Research Institute. Search theory and
applications, (New York: Plenum press, 1980).

Stanoyevitch, Alexander. Introduction to MATLAB with
numerical preliminaries, (Hoboken, N.J.: Wiley-
Interscience, 2005).

Washburn, Alan R. Search and detection, 4th ed., (United
States of America: Informs, 2002).

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1.  Defense Technical Information Center
    Ft. Belvoir, Virginia

2.  Dudley Knox Library
    Naval Postgraduate School
    Monterey, California

3.  James N.Eagle
    Naval Postgraduate School
    Monterey, California

4.  Sung Jin Kang
    Korea National Defense University
    Seoul, Republic of Korea

5.  Jae Young Lee
    Korea National Defense University
    Seoul, Republic of Korea

6.  Geon Wook Jeon
    Korea National Defense University
    Seoul, Republic of Korea